Collider-Accelerator Department
Brookhaven National Laboratory, Brookhaven Science Associates
Upton, New York 11973

V210 Datacon Master VME Module
User Manual
(v210Manual.doc)

October 2004
R. Michnoff

1.0 Overview

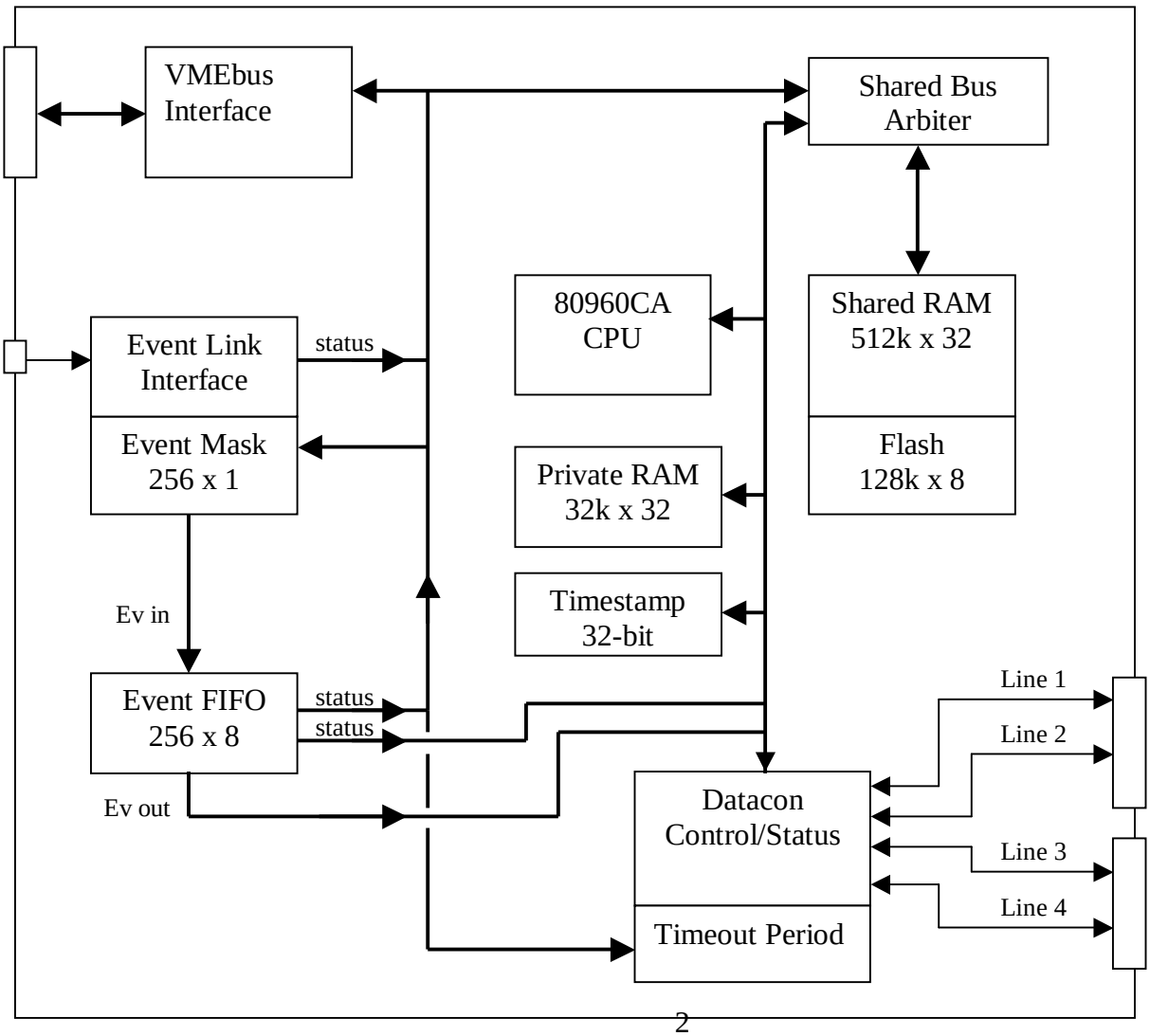
This document describes the V210 Datacon Master VME module. An operational description and detailed register definition is provided.

The V210 is functionally equivalent to its predecessor V110. The following enhancements have been incorporated.

1. The obsolete ROM has been replaced with flash memory, providing the capability to download the i960 firmware program via the VME bus.
2. The VME bus interface has been developed in an Altera gate array, eliminating the obsolete VIC068 VME interface component.
3. The V210 hardware is contained on a single VME module. The V110 required a VME module, buffer module and transition module.

2.0 Hardware Block Diagram

Following is a block diagram of the V210 hardware module.



3.0 VME memory Map

The V210 VME memory and registers are mapped to A32 address space. The base address offset is selected via jumpers for VME address lines A31..A22. The total VME address space required for the V210 is 4 Mbytes.

Address (offset from base)	**Description**	**Access**	**Size**
0x000000-0x000ef	VME ID	rd/wr D8	64 bytes
0x000000-0x1ffff	Flash memory	rd/wr D8	128 Kbytes
0x100000	Interrupt level	rd/wr D8	1 byte
0x100001	Interrupt vector	rd/wr D8	1 byte
0x100002	Interrupt enable bit 0 – PIL available interrupt triggered by CPU bit 1 – VME command complete interrupt triggered by CPU bit 2 – CPU fail interrupt bit 3 – Event FIFO overflow interrupt bit 4 – Event link error interrupt	rd/wr D8	1 byte
0x100003	Interrupt status Same bit definition as Interrupt enable	rd D8	1 byte
0x100004	Enable flash write Set bit 0 to logic 1 to enable flash writes	rd/wr D8	1 byte
0x100005	Trigger CPU interrupt Write any value to trigger interrupt	wr D8	1 byte
0x100006	Event link PLL is locked bit 0 – logic 1 = PLL is locked logic 0 = PLL is not locked	rd D8	1 byte
0x100007	Event link clock fail status bit 0 – logic 1 = Event link no carrier logic 0 = Event link OK	rd D8	1 byte
0x100008	CPU fail status bit 0 – logic 1 = CPU failed logic 0 = CPU OK	rd D8	1 byte
0x100009	Event FIFO full status bit 0 – logic 1 = FIFO full logic 0 = FIFO not full	rd D8	1 byte
0x10000a	Event FIFO empty status bit 0 – logic 1 = FIFO empty logic 0 = FIFO not empty	rd D8	1 byte
0x10000b-0x10000f	Unused		5 bytes
0x100010-0x100017	Scratch registers	rd/wr D8	8 bytes
0x100018-0x10001f	Unused		8 bytes
0x100020	Scratch register	rd/wr D8	1byte
0x100021-0x10002f	Unused		15 bytes
0x100030	Datacon Line 2 timeout 1 count = 2 microseconds	rd/wr D8	1byte

Address (offset from base)	**Description**	**Access**	**Size**
0x100031	Datacon Line 1 timeout 1 count = 2 microseconds	rd/wr D8	1byte
0x100032	Datacon Line 3 timeout 1 count = 2 microseconds	rd/wr D8	1byte
0x100033	Datacon Line 4 timeout 1 count = 2 microseconds	rd/wr D8	1byte
0x100034-0x17ffff	Unused		
0x180000-0x1800ff	Event FIFO event mask When bit 0 is set to logic 1, the detected event code will be sent to the i960 event fifo. The sub address 00-ff corresponds to the event code.	rd/wr D8	256 bytes
0x200000-0x3fffff	Shared RAM This memory area is mapped to both VME address space and i960 address space.	rd/wr D32	2 Mbytes
0x200000	Pending Intrpt List (PIL) entry index Written by i960, read by VME Valid values: 0-128	rd D32	4 bytes
0x200004	Pending Intrpt List (PIL) exit index Valid values: 0-128	rd/wr D32	4 bytes
0x200008	Millisecond counter This memory location is incremented by the i960 increments on each occurrence of the 1 millisecond interrupt. This is useful as a diagnostic to indicate that the i960 is alive.	rd D32	4 bytes
0x20000c	VME command A command is sent to the i960 by writing one of the following command codes to this register, followed by a write to the 'Trigger CPU Interrupt' register (base + 0x100005) 1 – Execute the i960 initialization code 2 – Execute the memory test The i960 sets this memory location to 0 when the command execution is complete.	rd/wr D32	4 bytes
0x200010	Fault flag A value of 0xf9f9 indicates that the i960 detected a fault condition.	rd D32	4 bytes
0x20002c	Memory test error count The number of errors that occurred during the last executed memory test.	rd D32	4 bytes
0x200068	Text buffer index The current index into the circular text buffer.	rd D16	2 bytes
0x200070	Text buffer The i960 writes ASCII characters to the text buffer during program execution. This provides a method to track the program execution path.	rd D8	1 Kbyte

Address (offset from base)	**Description**	**Access**	**Size**
0x200700	Maximum number of PPM users Valid values: 1-8	rd/wr D8	1 byte
0x200701	Maximum number of Datacon chans Valid values: 1-4	rd/wr D8	1 byte
0x20070c	Fiducial event code Typically T0 event code Valid values: 0-255	rd/wr D8	1 byte
0x20070d	User 1 event code Event codes for other users are assumed to sequentially follow the user 1 code. Valid values: 0-255	rd/wr D8	1 byte
0x200800	Pending Interrupt List (PIL) This is an array of 128 Pending Interrupt Block structures. The Pending Interrupt List structure is defined in this document.	rd/wr	1 Kbyte
0x200c00	Interrupt Request Blocks This is an array of 128 Interrupt Request Blocks (IRB). The IRB structure is defined in this document.	rd/wr	1 Kbyte

4.0 i960 Memory Map

Following is the address map definition for the i960 memory and registers.

Address	Description	Access	Size
0x10000000- 0x101fffff	Shared RAM This memory area is mapped to both VME address space and i960 address space.	rd/wr D32	2 Mbytes
0x10000000	Pending Intrpt List (PIL) entry index Written by i960, read by VME Valid values: 0-128	rd/wr D32	4 bytes
0x10000004	Pending Intrpt List (PIL) exit index Valid values: 0-128	rd D32	4 bytes
0x10000008	Millisecond counter This memory location is incremented by the i960 increments on each occurrence of the 1 millisecond interrupt. This is useful as a diagnostic to indicate that the i960 is alive.	rd/wr D32	4 bytes
0x1000000c	VME command VME sends a command to the i960 by writing one of the following command codes to this register, followed by a write to the 'Trigger CPU Interrupt' register (vme base + 0x100005) 1 – Execute the i960 initialization code 2 – Execute the memory test The i960 sets this memory location to 0 when the command execution is complete.	rd/wr D32	4 bytes
0x10000010	Fault flag A value of 0xf9f9 indicates that the i960 detected a fault condition.	wr D32	4 bytes
0x1000002c	Memory test error count The number of errors that occurred during the last executed memory test.	wr D32	4 bytes
0x10000068	Text buffer index The current index into the circular text buffer.	wr D16	2 bytes
0x10000070	Text buffer The i960 writes ASCII characters to the text buffer during program execution. This provides a method to track the program execution path.	wr D8	1 Kbyte
0x10000700	Maximum number of PPM users VME writes this value. Valid values: 1-8	rd D8	1 byte
0x10000701	Maximum number of Datacon chans VME writes this value. Valid values: 1-4	rd D8	1 byte
0x1000070c	Fiducial event code Typically T0 event code. VME writes this value. Valid values: 0-255	rd D8	1 byte
0x1000070d	User 1 event code VME writes this value. Event codes for other users are assumed to sequentially follow the user 1 code. Valid values: 0-255	rd D8	1 byte

Address	Description	Access	Size
0x10000800	Pending Interrupt List (PIL) This is an array of 128 Pending Interrupt Block structures. The Pending Interrupt List structure is defined in this document.	rd/wr	1 Kbyte
0x10000c00	Interrupt Request Blocks This is an array of 128 Interrupt Request Blocks (IRB). The IRB structure is defined in this document.	rd/wr	1 Kbyte
0x50000000	CPU run LED Write any value to flash front panel LED.	wr D8	1 byte
0x60000000	Datacon Line 1	rd/wr D32	4 bytes
0x60000004	Datacon Line 2	rd/wr D32	4 bytes
0x60000008	Datacon Line 3	rd/wr D32	4 bytes
0x6000000c	Datacon Line 4	rd/wr D32	4 bytes
0x60000010	Datacon Line 1 Status bit 0 - /dav_gat data avail – logic 1 when data avail bit 1 - timeout – logic 1 when timeout bit 2 – parity error – logic 1 when parity error	rd D32	4 bytes
0x60000014	Datacon Line 2 Status Same bit definition as Line 1 Status	rd D32	4 bytes
0x60000018	Datacon Line 3 Status Same bit definition as Line 1 Status	rd D32	4 bytes
0x6000001c	Datacon Line 4 Status Same bit definition as Line 1 Status	rd D32	4 bytes
0x60000020	Datacon Line 1 Reset Write any value to reset line.	wr D32	4 bytes
0x60000024	Datacon Line 1 Reset Write any value to reset line.	wr D32	4 bytes
0x60000028	Datacon Line 1 Reset Write any value to reset line.	wr D32	4 bytes
0x6000002c	Datacon Line 1 Reset Write any value to reset line.	wr D32	4 bytes
0x70000000	Trigger PIL available interrupt to VME Write any value to trigger interrupt.	wr D8	1 byte
0x70000001	Trigger command complete interrupt to VME Write any value to trigger interrupt.	wr D8	1 byte
0x70000002	Reset sysfail flip-flop after power up Write any value to reset sysfail.	wr D8	1 byte
0x80000000	Timestamp data (microseconds)	rd D32	4 bytes
0x80000000	Reset timestamp Write any value to reset timestamp to 0.	wr D32	4 bytes
0x90000000	Read Event code from FIFO	rd D8	1 byte
0x90000001	Event FIFO empty status bit 0 – logic 0 = not empty logic 1 = empty	rd D8	1 byte

Address	Description	Access	Size
0x90000002	Event FIFO empty status bit 0 – logic 0 = not empty logic 1 = empty	rd D8	1 byte
0x90000003	Event FIFO full status bit 0 – logic 0 = not full logic 1 = full	rd D8	1 byte
0x800000004	Clear Event FIFO Write any value to clear FIFO.	wr D8	1 byte
0xa0000000- 0xa001ffff	Private RAM	rd/wr d32	128 Kbytes
0xfffe0000-0xffffffff	Flash Memory This memory region is also mapped to VME.	rd/wr d8	128 Kbytes

5.0 Shared Memory Communications Protocol

(This section was edited from a document written by R. Warkentien, 11/1994, version 1.6)

In an attempt to be fast, concise, simple and flexible a communications protocol involving short blocks of information in a common memory bank has been devised.
Extensive use is made of pointers and linkages. The allocation of memory space is left to an off-board manager and the Datacon Engine will not check for incompatibilities.

5.1 Information Block Description

The protocol defines four types of information blocks:

1. Message Descriptor Blocks – contain information describing the size and location of a Source Message Block and one or more Reply Message Blocks.
2. Source Message Blocks – contains the Datacon images to be sent
3. Reply Message Blocks – contains the received Datacon image, time and status info
4. Input Request Blocks

Message Descriptor Blocks are added to the Engine's Trigger Dispatch Table using Input Request Blocks. These blocks define the trigger event and provide a pointer to the Message Descriptor Block to be executed upon occurrence of that event. Since more than one Input Request Block may request the same trigger event, a system of linked lists is implemented via pointers in the Message Descriptor Block.

Message Descriptor Blocks are dynamic in that they may be limited to one occurrence, retriggered endlessly or chained together in complicated patterns.

Input Request Block (IRB)

Request	byte	(bit0: 1=Interrupt, 0=Poll; bit1: 1=Insert, 0=Delete; bit6: 1=Failure, 0=Success, bit7: 1=Pending, 0=Done)
PPM User Number	byte	(0-8; 0 = non-PPM)
Time Event Code	byte	(0-255; 0 = ASAP)
D.C. Line Number	byte	(1-4)
MDB Pointer	word	(NULL pntr = delete)

A block of 100h words in shared memory is reserved for Input Request Blocks (IRB). The IRB contains a pointer to a Message Descriptor Block (MDB), which the requestor has previously created in shared memory; it also describes the trigger event that will call this MDB into action – user number, event code and line number. The user may request that an **interrupt** be created when the IRB is taken, or the Request byte may be **polled** to test for completion. The user marks the IRB **pending** when ready; the Datacon Engine marks/clears **failure** as appropriate and clears **pending** to indicate done. Once marked **done**, an IRB is ignored until again marked **pending**. Note that MORE THAN ONE IRB may refer to the same trigger event.

Message Descriptor Block (MDB)

Link Backward Pointer	word	(reserved)
Link Forward Pointer	word	(reserved)
Subsequent MDB pointer	word	(NULL pntr = only 1 Self pntr = repeat Valid pntr = chain)
SMB Element Count	short	(1-16384)
RMB Repeat Count	byte	(0-15)
RMB Repeat Index	byte	(reserved)
SMB Pointer	word	NULL pntr = IRB)
RMB Pointer(s)	word(s)	(RptCnt+1 pointers)

The MDB in its simplest form contains a count of the number of Datacon elements in the Source Message, a pointer to the Source Message Block (SMB), a pointer to the Reply Message Block (RMB) and a pointer which denotes which MDB will be used on Subsequent trigger events. Thus one can create a situation where:

1. The MDB is self canceling (NULL pointer)
2. The MDB retriggers forever (pointer to self)

The MDB can also accommodate certain special cases that arise from time to time. By specifying a non-zero Repeat Count, the user may specify several reply (RMB) pointers within a single MDB (multiple buffering). The same source (SMB) is used throughout and the Subsequent MDB pointer only comes into play when the requisite number (Repeat Count + 1) of triggers have occurred.

As noted above, the Subsequent MDB Pointer creates a mechanism whereby subsequent trigger event handling is manipulated. This pointer may point to any valid MDB so that a complex sequence is possible, if required.

Another special case involves the creation of sequential triggers, as complex trigger events cannot be handled directly. Inserting a NULL pointer for the SMB Pointer and using the RMB Pointer to point to a pseudo-IRB creates an equivalent construct, however (A true IRB is located at an assigned location in memory and is scanned in background; thus it is not useful for this application.) In fact the Repeat feature could be applied as well to create extremely complex sequences.

The Repeat Index and Linkage pointers are reserved for the Datacon Engine use.

Source Message Block (SMB)

Mode	word	(undefined)
{ D.C. Message (n)	word }	(D.C. transmit list)

The SMB contains some number (SMB Element Count as defined in the MDB) of Datacon images to be transmitted on occurrence of the appropriate trigger. The Mode word preceding the images is an as yet undefined quantity reserved for future expansion.

Reply Message Block (RMB)

	Start Time Stamp	word	
	End Time Stamp	word	
{	D.C. Reply	word	
	Status	short	
	Differential Time Stamp	short }	(repeat per Element Count)

The RMB is a buffer area where the Datacon replies are stored. Each reply includes 16 status bits and a 16-bit differential time stamp. In addition, a 32-bit time stamp is recorded upon transmission of the first SMB image and receipt of the last reply.

RMB size in words = 2 * SMB_Element_Count + 2

A word about time stamps: The start and end time stamps are a count of the number of microseconds since the last fiducial time event. Differential time stamps consist of the low order 16 bits of the true time stamp and must be referenced with the Start/End times to compute a real time.

5.2 Installing Message Descriptor Blocks

Datacon Trigger Dispatch Table

The trigger dispatch table is a 3 dimensional array of MDB pointers as shown below. The MDB pointed to is the first to be executed in the linked list. This table provides a unique entry for each PPM user, event code and datacon line. PPM user index 0 is used for immediate command lists.

{	PPM User Number	(8+1)	
	Time Event Code	(255)	
	D.C. Line Number	(4)	
	MDB Pointer	}	

In order to successfully insert a timed Datacon request into the queue, the user must:

1. Secure exclusive use of sufficient memory space for both the SMB(s) and RMB(s).
2. Fill in the SMB(s) with Datacon images.
3. Secure and prepare the MDB(s).
4. Secure and prepare an IRB.

The Datacon Engine will insert the MDB Pointer at the appropriate location within the Trigger Dispatch Table and signal completion by setting **done** in the IRB Request byte (and interrupt the user if requested). If a MDB pointer already occupies the desired trigger location in the Dispatch Table, a linked list is created using the Link Forward, and Link Backward pointers in the MDB. This technique allows MDBs to be dynamically added or deleted with minimal disruption.

The Subsequent MDB Pointer in the Message Descriptor Block is used to modify the MDB linked list following a Trigger Event as though an IRB were executed. When a Subsequent MDB Pointer is not NULL and does not point to itself, then after execution the MDB pointed to by the Subsequent MDB is added to the linked list and the current MDB is deleted from the linked list. By chaining MDBs, the user may create any number of Reply structures.

5.3 Use of NULL pointers

A NULL pointer may be used to delete requests when applied to the MDB Pointer. In the case of the Subsequent MDB Pointer, NULL indicates "once only" operation. However, in the case of the Input Request Block's MDB Pointer, NULL will cancel ALL MDBs linked to that trigger.

A NULL pointer inserted for the RMB Pointer indicates that NO REPLY MESSAGE is expected (but no status or time stamp is returned either).

A NULL SMB pointer is a special case denoting that the RMB Pointer actually points to an Input Request.

5.4 Interrupt Handling

In order to ascertain when Reply Message Blocks are ready for processing, an interrupt may be generated for every RMB that is filled. The VME interrupt level and vector are configurable. Reading the Pending Interrupt List identifies an interrupt. Each event produces a two-word entry in the list:

1. The trigger event descriptor (RMB#, user#, event#, line#)
2. The MDB pointer

When configured to interrupt on IRB completion, a similar event descriptor will be generated (0xFF, user#, event#, line#) followed by the IRB pointer.

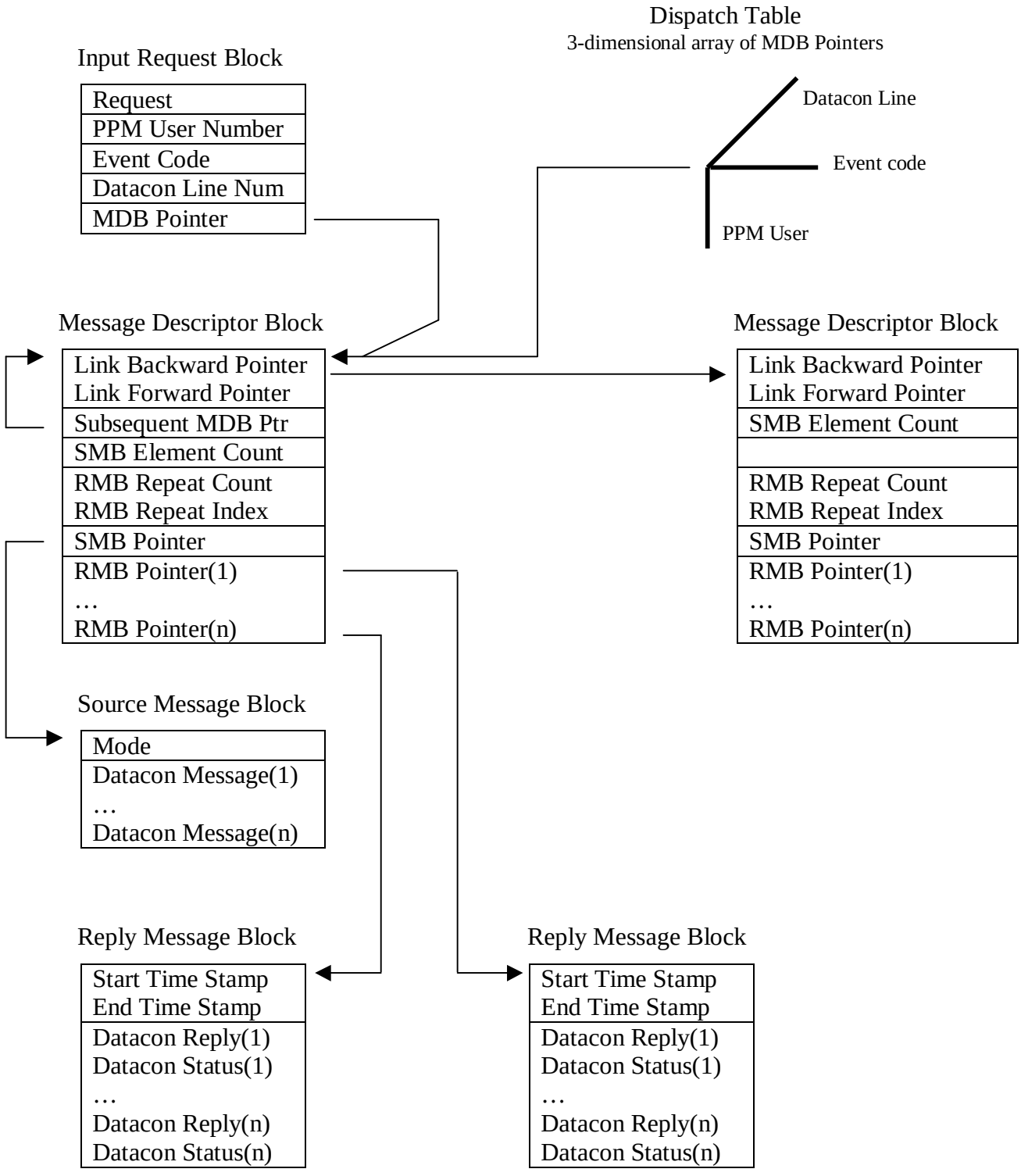
The possibilities for confusion regarding interrupt identification are many; therefore the Pending Interrupt List is implemented as a 100h word circular buffer, where the following shared memory locations are used to track the entry and extraction indexes.

A32 base + 0 pending entry index (4-byte value)
A32 base + 4 pending exit index (4-byte value)

Pending Interrupt List Structure

RMB Index	byte	
PPM User Number	byte	(0-8; 0 = non-PPM)
Time Event Code	byte	(0-255; 0 = ASAP)
D.C. Line Number	byte	(1-4)
MDB Pointer	word	

6.0 Data Structure Diagram



7.0 Drawing Numbers

D09-2979	V210 Datacon Master Programmed Assembly
D09-2978	V210 Datacon Master Module Assembly
D09-2977	V210 Datacon Master Front Panel Detail
D09-2976	V210 Datacon Master Printed Wiring Board Assembly
D09-2975	V210 Datacon Master Printed Wiring Board Drilling
D09-2974	V210 Datacon Master Schematic
94028874	Phase Lock Loop Event Link Decoder Programmable Device